IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

JUXTACOMM TECHNOLOGIES, INC.,	§	
	§	
Plaintiff	§	
	§	
vs.	§	**CASE NO. 2:07CV359**
	§	**PATENT CASE**
ASCENTIAL SOFTWARE CORPORATION, BUSINESS OBJECTS SA, BUSINESS OBJECTS AMERICA, CA, INC., COGNOS CORPORATION, DATAMIRROR, INC., FIORANO SOFTWARE, INC., HUMMINGBIRD LTD., INTERNATIONAL BUSINESS MACHINES CORPORATION, INFORMATICA CORPORATION, INFORMATION BUILDERS, INC., METASTORM, INC., MICROSOFT CORPORATION, OPEN TEXT CORPORATION, SOFTWARE AG, INC., SYBASE, INC., WEBMETHODS, INC., and INTERSYSTEMS CORPORATION,	§	
	§	
Defendants		

MEMORANDUM OPINION

This Memorandum Opinion construes the disputed terms in U.S. Patent No. 6,195,662 ("the '662 patent"), entitled "System for Transforming and Exchanging Data Between Distributed Heterogeneous Computer Systems." The Court previously issued a preliminary order construing the claims of the '662 patent. This claim construction opinion explains those constructions.

BACKGROUND

The '662 patent concerns the exchange of data between computer systems operating on the basis of different data formats. The '662 patent discloses use of a software-based system that transforms data structures imported from a source computer system into data structures compatible

with and for export to a target computer system. The patent contains three independent claims: system claim 1, method claim 13, and system claim 17.

APPLICABLE LAW

"It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude.'" *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc) (quoting *Innova/Pure Water Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1115 (Fed. Cir. 2004)). In claim construction, courts examine the patent's intrinsic evidence to define the patented invention's scope. *See id.*; *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 861 (Fed. Cir. 2004); *Bell Atl. Network Servs., Inc. v. Covad Commc'ns Group, Inc.*, 262 F.3d 1258, 1267 (Fed. Cir. 2001). This intrinsic evidence includes the claims themselves, the specification, and the prosecution history. *See Phillips*, 415 F.3d at 1314; *C.R. Bard, Inc.*, 388 F.3d at 861. Courts give claim terms their ordinary and accustomed meaning as understood by one of ordinary skill in the art at the time of the invention in the context of the entire patent. *Phillips*, 415 F.3d at 1312–13; *Alloc, Inc. v. Int'l Trade Comm'n*, 342 F.3d 1361, 1368 (Fed. Cir. 2003).

The claims themselves provide substantial guidance in determining the meaning of particular claim terms. *Phillips*, 415 F.3d at 1314. First, a term's context in the asserted claim can be very instructive. *Id*. Other asserted or unasserted claims can also aid in determining the claim's meaning because claim terms are typically used consistently throughout the patent. *Id*. Differences among the claim terms can also assist in understanding a term's meaning. *Id*. For example, when a dependent claim adds a limitation to an independent claim, it is presumed that the independent claim does not include the limitation. *Id*. at 1314–15.

"[C]laims 'must be read in view of the specification, of which they are a part.'" *Id*. (quoting

Markman v. Westview Instruments, Inc., 52 F.3d 967, 979 (Fed. Cir. 1995) (en banc)). "[T]he specification 'is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id*. (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)); *Teleflex, Inc. v. Ficosa N. Am. Corp.*, 299 F.3d 1313, 1325 (Fed. Cir. 2002). This is true because a patentee may define his own terms, give a claim term a different meaning than the term would otherwise possess, or disclaim or disavow the claim scope. *Phillips*, 415 F.3d at 1316. In these situations, the inventor's lexicography governs. *Id*. Also, the specification may resolve ambiguous claim terms "where the ordinary and accustomed meaning of the words used in the claims lack sufficient clarity to permit the scope of the claim to be ascertained from the words alone." *Teleflex, Inc.*, 299 F.3d at 1325. But, "'[a]lthough the specification may aid the court in interpreting the meaning of disputed claim language, particular embodiments and examples appearing in the specification will not generally be read into the claims.'" *Comark Commc'ns, Inc. v. Harris Corp.*, 156 F.3d 1182, 1187 (Fed. Cir. 1998) (quoting *Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1571 (Fed. Cir. 1988)); *see also Phillips*, 415 F.3d at 1323. The prosecution history is another tool to supply the proper context for claim construction because a patent applicant may also define a term in prosecuting the patent. *Home Diagnostics, Inc., v. Lifescan, Inc.*, 381 F.3d 1352, 1356 (Fed. Cir. 2004) ("As in the case of the specification, a patent applicant may define a term in prosecuting a patent.").

Although extrinsic evidence can be useful, it is "'less significant than the intrinsic record in determining the legally operative meaning of claim language.'" *Phillips*, 415 F.3d at 1317 (quoting *C.R. Bard, Inc.*, 388 F.3d at 862). Technical dictionaries and treatises may help a court understand the underlying technology and the manner in which one skilled in the art might use claim terms, but technical dictionaries and treatises may provide definitions that are too broad or may not be

indicative of how the term is used in the patent. *Id*. at 1318. Similarly, expert testimony may aid a court in understanding the underlying technology and determining the particular meaning of a term in the pertinent field, but an expert's conclusory, unsupported assertions as to a term's definition is entirely unhelpful to a court. *Id*. Generally, extrinsic evidence is "less reliable than the patent and its prosecution history in determining how to read claim terms." *Id*.

DISPUTED TERMS

Script and **Script processor**

Although the parties briefed these terms separately, Defendants seek the same limitations, for the same reasons, in both terms. Accordingly, the Court will address them together. JuxtaComm contends a "script" is "a group of commands to control data movement into and out of the system, and to control data transformation within the system." Defendants contend a "script" is "a series of text commands interpretively run by the script processor, such that one command at a time is translated and executed at runtime before the next command is translated and executed, and that control data movement into and out of the system and control data transformation within the system." Both sides agree that scripts are commands that control data movement into and out of the system and control data transformation within the system. Defendants seek to further limit "script" to text commands as opposed to graphical commands. Defendants also seek to limit scripts to being interpretively run by translating and executing one command at a time.

JuxtaComm contends the Court should construe "script processor" as "software component that processes a script." Defendants contend the Court should construe the term as "a software component that interpretively runs scripts by translating and executing one script command at runtime, before translating and executing the next script command." Both sides agree a script processor is a software component that processes or runs scripts, but Defendants seek to add the

same additional limitations to the construction as they sought with "script."

The specification expressly defines "script": "Scripts 55 must be defined to control data movement into and out of the system, and to control data transformation within the system." Col. 5:65–67. Defendants seek to further limit scripts to text commands and to being interpretively run by translating and executing one command at a time.

Defendants contend script should be limited to textual commands because graphical or visual scripts are not described in the patent and all of the examples in the patent are of text commands. The claims are not limited to textual scripts, and the Court will not import implied limitations from the specification when the claims are not so limiting. Defendants are correct that the '662 patent does not specifically disclose graphical or visual scripts. While all of the preferred embodiments described in the '662 patent are textual scripts, nowhere in the '662 patent or its claims does the inventor disavow coverage of other scripts. Defendants are unable to point to any place in the specification or prosecution history where the patentee clearly disavowed such coverage. Accordingly, the Court will not import the limitations of the preferred embodiments into the claims, and the Court rejects Defendants' limitation that would limit scripts to textual commands.

Defendants argue that scripts should be limited to being interpretively run by translating and executing one command at a time, rather than being compiled. Defendants contend that this is the ordinary meaning of script and that any other meaning was disclaimed during prosecution to overcome a rejection in light of *Morgenstern*.

In the specification, a script is one item of metadata in the metadata database. Col. 4:32-39. A script is "run" on the script processor. Col. 4:41-43. There is no indication in the written description of a requirement for running a script using an interpreter; nor is there any indication that a script in the metadata database cannot be compiled by the script processor. A script is indicated

as being nothing more than a task oriented program (batch file set of commands) that controls data transformation by the rules processor and the movement of data. The specification does not speak to exactly how the script is being "run." In general, a script is a program that is executed by another program rather than being directly run on the hardware processor. Accordingly, the specification does not limit "scripts" to being interpretively run.

As originally filed, the claims recited "a script processor for controlling data transformation . . . ." The examiner rejected the claims as being obvious over *Morgenstern* and *Mitchell*. In making the rejection of claim 2, which is claim 1 of the issued '662 patent, the examiner confusingly referred to "browser," while citing to the disclosure in *Morgenstern* of the information bridge mediator 60, as being a "script processor" for controlling data transformation. *See* Office Action April 3, 2000 at 3, ¶ 6. *Morgenstern* is diagramed as:

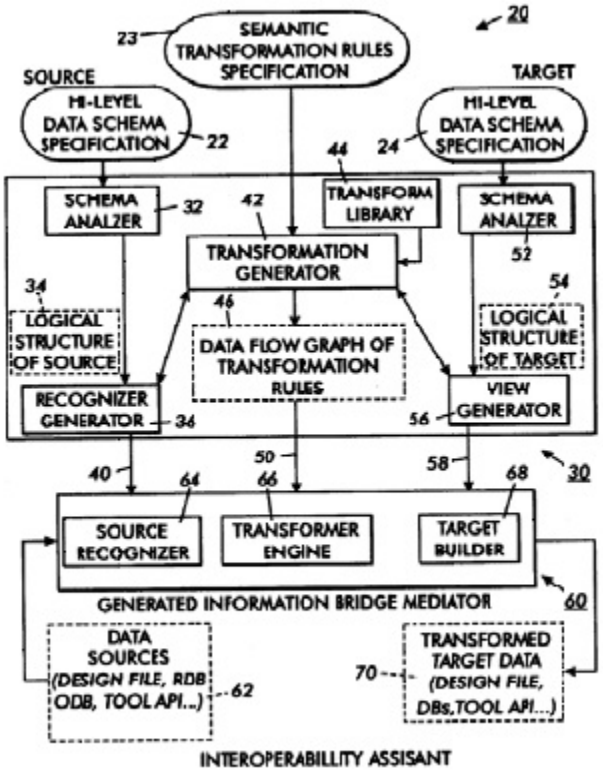

FIG. 2

What *Morgenstern* describes is data transformation using transformer engine 66 of bridge 60, which is a compiled program of transformation rules that are executed on the computer processor. The code for bridge 60 is built during a system generation phase. The source and target data structure specifications are applied to a transformation generator 42, which generates code files that describe the transformation rules to be applied. The code is then compiled for running on the computer processor. *Morgenstern* does not have a batch file that is pulled from a database during run time to control the transformation.

In response to the obviousness rejection, the applicants amended claims 2 and 18 to recite that the script processor "utilizes metadata" to control data transformation. They generally characterized *Morgenstern* as describing a generation module 30 which outputs a compiled information bridge 60 adapted to convert data from a source format to a target format. They then argued that *Morgenstern* teaches a different methodology.

The applicants also made separate arguments in regard to claims 2 and 18. The arguments in regard to claim 18, which recites "executable code for providing a script processor . . . to control data transformation," were that metadata (i.e., a script) stored in a database is utilized to "interpretively convert data from a source to a target." This statement does not say "interpretively convert the metadata (script)." In context, the applicants' argument was not making a distinction between "compiling" and "interpreting," but rather it was indicating that there is use of a sequence of commands (scripts generated by a user) extracted from a database that are processed by a script processor program before execution by the computer processor rather than being executed directly by the computer processor. In arguments related specifically to claim 2, the applicants reference the absence in both *Morgenstern* and *Mitchell* of "scripts or a scripts processor." 9/05/2000 Response at 7. In other words, in contrast to *Morgenstern*, a sequence of commands pulled from a database

is being carried out by another program (script processor) rather than by the computer processor.

The prosecution history does not evidence a clear restriction of "script processor" to a program that is interpreting a script for execution rather than compiling a script for execution. The distinction that was made over *Morgenstern* is that of using an intermediate program (script processor), whether it is an interpreter or a compiler, that executes a script program pulled from a database. The compiler program in *Morgenstern*, col. 8:58-61, could be labeled a script processor. However, the automatically generated code that is compiled, even if labeled a "script," is produced by the transformer generator 42 and is not pulled from a database. Thus, even if the transformer generator 42 would be labeled a "configuration management user interface," absent from *Morgenstern* would be a database storing scripts. Finally, if "interpretively run" was necessary and being used to distinguish *Morgenstern*, the examiner would have required the claims to be amended to state that limitation. Accordingly, the Court will not limit "script" based on the prosecution history.

Accordingly, the Court adopts JuxtaComm's proposed construction and construes "script" as "a group of commands to control data movement into and out of the system, and to control data transformation within the system." The Court also adopts JuxtaComm's proposed construction and construes "script processor" as "software component that processes a script."

Rule

JuxtaComm argues "rule" means "one or more statements." Defendants contend "rule" means "one or more statements that are executed from top to bottom to perform a specific operation to achieve a desired result." Thus, the parties dispute whether rules must be "executed from top to bottom to perform a specific operation to achieve a desired result."

The specification states:

> The purpose of a rule is to perform a specific operation to achieve a desired result. A rule is one or more statements. These statements are executed from top to bottom and when the last statement within the rule has been executed, or an Exit statement is encountered, the rule ends.

Col. 5:22–27. The additional language that Defendants seek to include in the construction concern the purpose of a rule and how the rule statements are executed. Those aspects of the disclosure do not further define what a rule *is*. Additionally, those limitations would exclude disclosed embodiments that use conditional processing or looping. *See* Fig. 9; col. 7:33–35; 8:54–60. Accordingly, the Court does not include those limitations in its construction. The Court construes "rule" as "one or more statements."

Data transformation rule sets

JuxtaComm proposes that the Court construe "data transformation rule sets" as "a collection of rules for transforming data." Defendants propose "a collection of rules used to transform a data bag in one format into another data bag of a different format."

To support their additional limitations, Defendants rely on the specification, which they contend defines the term: "Rule sets 54 (Fig. 5) are collections of rules within the present invention. Rule sets are used to transform a data bag in one format into another data bag of a different format." Col. 5:20–22. Defendants also rely other parts of the specification, which they contend describe the data transformation rule set as operating on the data bags. *See* col. 7:7–9, 8:29–32, 8:54–55. Finally, Defendants contend that the claims require that the "rule set processor" is "for manipulating a data bag for storing imported data and a data bag for storing exported data." *See* claim 1.

Similar to "rule," the definition in the specification only specifies that the term means "a collection of rules." The additional statements express what the rules are used to do but do not further define what the rules are and are therefore not properly part of the definition. As the specification does not expressly limit the claim language, the claim language is controlling. The

claim language does not require that these rule sets operate on the data bags, but on the data. Had the patentees intended to claim otherwise, they could have claimed "data bag transformation rule sets." That the rule set processor, a separate claim element, is "responsive to said script processor for manipulating a data bag for storing imported data and a data bag for storing exported data" does not affect the meaning of "data transformation rule sets."

Accordingly, the Court construes "data transformation rules sets" as "a collection of rules for transforming data."

Utilizing metadata from the metadata database

JuxtaComm contends this term does not require construction or should be construed as "using metadata from a metadata database." JuxtaComm contends its construction is supported by the plain and ordinary meaning. In their briefing, Defendants contended the term should be construed to mean "accessing the logical import and export data interfaces and data transformation rule sets from the metadata database during execution of a script." At the hearing, Defendants offered the following modified construction: "accessing the logical import and export data interfaces and data transformation rule sets *and scripts* from the metadata database during execution of *the script processor*." Defendants contend their construction is required by the prosecution history.

According to Defendants, the applicants argued that accessing metadata while a script is being executed allows the source and target data structures to be modified without having to regenerate the entire script. This allows the claimed invention to be more flexible than *Morgenstern*. Also, according to Defendants, the applicants argued that in *Morgenstern* the transformation was performed without accessing the metadata, whereas the '662 patent uses instructions in the script to identify and access metadata items when the script is executed.

The Court disagrees with Defendants' characterization of the prosecution history. The entire

phrase indicates that nothing more than a script is being accessed during run-time. The primary distinction being made by the applicants in their remarks was that *Morgenstern* does not disclose a script processor that accesses metadata (i.e., a script) from the database when data transformation is to occur. That is, the script processor executes a script stored in the metadata database to convert data. Instead, *Morgenstern* uses a pre-built transformer engine 66 that is activated when data transformation is to occur. In other words, there is no "intermediate" program structure that converts the script commands to executable code. There being no such structure in *Morgenstern*, there necessarily is no metadata database to access to load a script when data transformation is to occur.

The specification describes that: "Scripts 55 must be defined to control data movement into and out of the system, and to control data transformation within the system." Col. 5:65-67. Thus, scripts are identified as the metadata being "utilized." The requirement that the script processor access the database for data transformation is inherent in the claim language itself ("utilizing metadata . . . to control data transformation within said systems interface . . . "). Thus, when data transformation is to occur, a script from the database is accessed for use. Col. 7:19-20 ("Finally, in step 80, the user initiates the script processor 37 to execute the script."). The prosecution history does not support Defendants' version of the limitations imposed by the applicants' remarks. Rather than indicating that the phrase means that the metadata items are accessed by the script processor during execution of a script, the remarks only point out that *Morgenstern* does not use a script of commands that is pulled from a database and converted by a script processor program to a form for execution by a computer processor.

Defendants' construction also is inconsistent with the specification. Defendants' construction has the script processor accessing the data transformation rule sets, whereas the specification and claim 9 indicate that the rule set processor accesses the database to obtain the data

transformation rule sets. *See* col. 4:32–37; 4:45–48; 7:16–18.

Finally, Defendants' proposed construction is inconsistent with the claim language. The claim indicates nothing more than that the script processor utilizes script metadata obtained from the database during data transformation. Defendants' construction, however, indicates that other metadata (data interfaces and data transformation rule sets) are accessed.

Accordingly, the Court construes the term as "using metadata from a metadata database."

Systems interface

JuxtaComm contends that "systems interface" does not require construction or, if construed, means "an interface to the distribution system." Defendants contend "systems interface" means "a component that enables a user to interact with the system." Defendants argue that "systems interface" is a user interface because the claimed functions performed by the systems interface are ones that the patent teaches are performed by a user.

Claim 1 does not expressly mention a user. In claim 2, the applicants expressly claimed a user interface. Thus, the applicants claimed a user when they intended a user to be a claim element. The claim 1 systems interface does not inherently require a user as it could be coupled to another computer system rather than a user terminal. *See* col. 7:19–22. Accordingly, the Court will not add a user limitation to claim 1. The Court construes "systems interface" as "an interface to the distribution system."

Rule set processor responsive to said script processor

JuxtaComm argues "rule set processor" should be construed to mean "software that processes rule sets." Defendants contend "rule set processor" should be construed as "a software component invoked by the script processor to execute rule sets." Although the parties briefed the term "rule set processor," at the claim construction hearing they agreed the dispute includes the meaning of

"responsive to said script processor." Accordingly, the Court will address and construe the entire phrase.

Defendants contend the rule set processor is an entirely separate component of the system. Defendants also seek to replace "responsive to," used in the claims, with "invoked by," used in their construction. Defendants seek to improperly limit the term to the disclosed embodiments and characterizations of the written description rather than what is recited in the claim.

Although the claim sets forth "script processor" and "rule set processor" as separate claim limitations, such a recitation does not require them to be separate physical components as indicated by Defendants' proposed construction. Nothing in the intrinsic evidence indicates that the two processors, admitted to be software, cannot be folded together and implemented as a single module. While the system functional diagram of Fig. 2 relied upon by Defendants shows the processors as separate functional blocks, the depiction does not restrict an implementation of the system to separate components or modules.

As to Defendants' request to include "invoked by" in the claim construction in place of "responsive to," the contention is that the terms are synonymous. If that is so, as JuxtaComm says, there is no reason to replace the term "responsive to." Moreover, the use of the term "invoked" in dependent claim 9 suggests to one skilled in the art a difference in scope between the two terms regardless of the presence of other possible limiting recitations in the dependent claim that form a basis for claim differentiation.

Accordingly, the Court construes "rule set processor responsive to said script processor" as "software component that processes rule sets responsive to said script processor."

CONCLUSION

For the foregoing reasons, the Court interprets the claim language in this case in the manner

set forth above. The claims with the disputed terms in bold are set forth in Appendix A. For ease of reference, the Court's claim interpretations are set forth in Appendix B.

So ORDERED and SIGNED this 13th day of July, 2009.

LEONARD DAVIS
UNITED STATES DISTRICT JUDGE

APPENDIX A

1. A **distribution system** for transforming and exchanging data between heterogeneous computer systems, comprising:

a) a **systems interface** for defining logical import and export data interfaces, **data transformation rule sets** and **scripts**;

b) a **metadata database** for storing said logical import and export data interfaces, **data transformation rule sets** and **scripts**;

c) a **script processor** for **utilizing metadata from the metadata database** to control data transformation within said **systems interface** and movement of said data into and out of **said distribution system**; and

d) a **rule set processor** responsive to said **script processor** for manipulating a **data bag** for storing imported data and a **data bag** for storing export data.

2. A distribution system as claimed in claim 1, wherein said **systems interface** comprises a configuration management user interface used by a user to define said logical import and export data interfaces, and create **data transformation rule sets** and **scripts**.

5. A **distribution system** as claimed in claim 4, wherein said import data view is used during execution of said **script processor** to load data from said import data source into said **import data bag**.

6. A **distribution system** as claimed in claim 3, wherein said logical export data interface is used to export data in said **data bag** out to an export data target.

7. A **distribution system** as claimed in claim 6, wherein export data view of said **export data bag** is used during execution of said **script processor** to save data from said **export data bag** out to said export data target.

8. A **distribution system** as claimed in claim 1, wherein defined **scripts** stored in said **metadata database** are executed by said **script processor**.

9. A **distribution system** as claimed in claim 1, wherein said **rule processor** is invoked by said **script processor** to transform the **import data bag** into the **export data bag** based on predefined data transformation **rules**.

10. A **distribution system** as claimed in claim 1 wherein said **script processor** makes use of a script control language for controlling data transformation within said **system interface** and movement of said data into and out of said **distribution system**.

11. A **distribution system** as claimed in claim 10, wherein said script control language comprises a set of script commands and a script command processor to process and execute each of a number of script command lines.

12. A **distribution system** as claimed in claim 11, wherein said set of script commands comprises a load command to load data into an **import data bag** from an import data connection; a sort command for sorting data in a **data bag** into a different order; a merge command for merging together data in a number of **data bags**; an append command for appending data from one **data bag** into another **data bag**; a copy command for copying one **data bag** into another **data bag**; a join command for joining two or more **data bags** into another **data bag**; a format command for formatting a **data bag** into another **data bag** using a defined **rule** set; and a save command for saving data from an **export data bag** out to an export data connection.

13. In a **distribution system** for transforming and exchanging data between heterogeneous computer systems, a method of controlling data transformation within said **distribution system**, comprising the steps of:

a) operating a **script processor** that **utilizes metadata** stored in a **metadata database** to control the loading of data into an **import data bag** from a logical import data interface and performing any one or more of the following steps to convert the data to a desired format in an **export data bag**;

1) sorting said data according to a predetermined order;

2) merging data from a number of data bags into one **data bag**

3) appending data from a first **data bag** into another **data bag** of the same type;

4) copying data from a first **data bag** into another **data bag** of the same type;

5) joining data from two or more **data bags** into another **data bag** using a specified key;

6) formatting data from a **data bag** into another **data bag** of a different type, using a defined **rule set**; and

b) saving the data in the **export data bag** out to an export data connection.

16. A method as claimed in claim 15 wherein said import data view is used during execution of a **script processor** to load data from said import data source into said **import data bag**.

17. A computer readable memory for transforming and exchanging datastore data between heterogeneous computer systems using different datastore formats for storing similar information, comprising:

a) executable code for providing a **systems interface** for defining logical import and export data interfaces, **data transformation rule sets** and **scripts**;

b) executable code for providing a **script processor** for **utilizing metadata from a metadata database** to control data transformation within said **systems interface** and movement of said data into and out of said **distribution system**; and

c) executable code for providing a **rule set processor** responsive to said **script processor** for manipulating a **data bag for storing imported data** and a **data bag for storing export data**.

18. A computer readable memory as claimed in claim 17 wherein the **metadata database** stores logical import and export data interfaces, **data transformation rule sets** and **scripts** executed by the **script processor**.

19. A computer readable memory as claimed in claim 17 further comprising a script control language used by the **script processor** to control data transformation within said **system interface** and movement of said data into and out of said **distribution system**.

APPENDIX B

Term or Phrase	Court's Construction
distribution system	[AGREED] a computer system for importing data from a source computer system, transforming the imported data and exporting the transformed data to a target computer system
systems interface	an interface to the distribution system
data transformation rule sets	a collection of rules for transforming data
rule	one or more statements
script	a group of commands to control data movement into and out of the system, and to control data transformation within the system
metadata database	[Agreed] a database that stores the logical import and export data interfaces, data transformation rule sets and scripts used by the system
script processor	software component that processes a script
utilizing metadata from the metadata database	using metadata from a metadata database
said distribution system	[AGREED] a computer system for importing data from a source computer system, transforming the imported data and exporting the transformed data to a target computer system
rule set processor responsive to said script processor	software component that processes rule sets responsive to said script processor
data bag	[AGREED] a data bag is stored in non-persistent memory, is created by the script processor and exists while the script is running, and contains both generic format data and definitions of that data
data bag for storing imported data	[AGREED] a data bag for storing imported data before it is transformed
data bag for storing export data	[AGREED] a data bag for storing export data after it is transformed
import data bag	[AGREED] a data bag for storing imported data before it is transformed
export data bag	[AGREED] a data bag for storing export data after it is transformed